

UM PROTÓTIPO DE BANCO DE DADOS DISTRIBUÍDOS – CASO UNOCHAPECÓ

PEREIRA, Monica Tisisani De Toni¹; PIVA, Rodrigo Tauffer²

¹ Prof^ª. Especialista em Informática; Orientadora; e-mail: monica@ unochapeco.edu.br

² Aluno do Curso de Bacharelado em Ciência da Computação; e-mail: piva@ unochapeco.edu.br

UM PROTÓTIPO DE BANCO DE DADOS DISTRIBUÍDOS – CASO UNOCHAPECÓ

RESUMO

Este trabalho apresenta um estudo sobre Sistemas Gerenciadores de Banco de Dados (SGBD), com maior ênfase em sistemas de banco de dados distribuídos, propondo a aplicação de seus conceitos para um caso real. Foram estudadas as técnicas de replicação e distribuição de dados, bem como os métodos de fragmentação horizontal, vertical e híbrida de dados, de tal forma a utilizar uma das opções para a distribuição e replicação de dados juntamente com os recursos do SGBD Oracle®.

O estudo de caso, iniciou-se com um problema encontrado na Universidade Comunitária Regional de Chapecó (UNOCHAPECÓ) e o Centro de Atendimento a Comunidade (CAC), que possui um programa computacional que serve de apoio aos atendimentos oferecidos pela instituição. Este mesmo deve acessar seus dados e replicá-los para uma base mestre, sem que haja perda de performance, bem como assegurar a integridade dos dados. Através dos estudos realizados, sugere-se a aplicação de uma técnica para a distribuição, alocação e replicação de dados ao longo da rede de computadores, afim de sugerir uma solução e deixar uma referência para trabalhos futuros que envolvam a mesma área.

Palavras-chave: SGBD, SGBDD, distribuição, fragmentação, replicação.

ABSTRACT

This work presents a study on Database Management Systems (DBMS), with greater emphasis in systems of distributed database, considering the application of its concepts for a real case. The techniques of response and distribution of data had been studied, as well as the methods of horizontal, vertical and hybrid fragmentation data, such forms together use one of the options for the distribution and response of data with the resources of the DBMS Oracle®.

The case study initiated with a problem found in the Regional Communitarian University of Chapecó (UNOCHAPECÓ) and the Community Center of Attendance (CCA) that possesses a computational program that serves of support to the attending offered for the institution. This one must access its data and replicate them for a data base master, without loss of performance, as well as assuring the integrity of the data. Through the carried through studies, it is suggested application of one technique for the distribution, allocation and response of data along with the computer network, similar to suggest one solution and to leave a reference for future works that involve the same area.

Key-words: DBMS, DDBMS, distribution, fragmentation, response.

1. INTRODUÇÃO

Atualmente existem à disposição do mercado várias estratégias para resolver e aplicar projetos computacionais dentro das organizações. Sabe-se que muitas empresas têm a tendência de crescer e espalhar suas filias por diversas partes do país ou até mesmo do exterior.

Com essa descentralização, os dados ficam distribuídos ao longo do conjunto pelo qual se forma a organização, sendo que, cada filial gera e necessita obter suas próprias informações,

tanto de cunho gerencial quanto administrativo, para tomada de decisão, planejamento de metas e definir estratégias administrativas.

Na UNOCHAPECÓ (Universidade Comunitária Regional de Chapecó) não poderia ser diferente, trata-se de uma instituição do ramo educacional, portanto, tem seus projetos de apoio à comunidade disseminados pela região oeste do estado de Santa Catarina, gerando assim, a distribuição de seus dados.

Considerando isso, as empresas que desenvolvem softwares para gerenciamento de dados produzem e comercializam novos produtos para atender a necessidade de seus clientes. E para resolver o problema da descentralização de dados, o mercado dispõem de sistemas gerenciadores de banco de dados distribuídos, os quais possibilitam que as organizações tenham seus dados distribuídos ao longo de uma rede de computadores, mas, sempre que necessário, conseguem obter informações dos diversos locais como se fossem um só.

Assim sendo, o presente trabalho apresenta conceitos sobre banco de dados e projeto de banco de dados, sistemas de bancos de dados distribuídos, arquitetura de SGBD distribuídos, projeto de banco de dados distribuído, e também as principais tecnologias para alocação de dados em bancos de dados distribuídos, através da implementação de um protótipo com base nas opções disponíveis no SGBD Oracle®. No decorrer deste trabalho explana-se como funcionam as regras de implementação da integridade dos dados, controle da concorrência, problemas relacionados com a distribuição de dados, bem como vantagens e desvantagens de se ter um sistema distribuído.

Esta pesquisa teve uma forte referência nas obras de Özsu (2001) e Dye (1999). Do primeiro foram estudadas as questões relativas as regras para se obter um sistema distribuído, onde são conceituadas e referenciadas algumas das abordagens das questões de fragmentação de dados e correção de fragmentação. Em Dye (1999), estudaram-se as opções para a distribuição e replicação de dados encontrados no SGBD Oracle®, de tal forma a ser explanada com maior ênfase a técnica de visões atualizáveis, que foi utilizada na implementação do modelo proposto.

Tratando-se de SGBD relacionais, foram conceituados e exemplificados os predicados para obter dados através da álgebra relacional e da *Structured Query Language* (SQL).

Baseado na bibliografia de Loney (1999), aborda-se a arquitetura cliente/servidor e em Heuser (2000) todos os conceitos e regras para se obter um banco de dados relacional, tratamentos de chaves primárias, estrangeiras e conceitos de integridade de dados.

Para o desenvolvimento do trabalho, primeiramente foram estudadas todas essas obras bibliográficas citadas e em seguida buscou-se um caso real para a aplicação da base teórica estudada no decorrer do período em que a pesquisa estava em andamento. Todos esses tópicos são tratados no decorrer dos capítulos. Apresentando os principais aspectos de banco de dados centralizados e distribuídos.

2. BANCO DE DADOS

Com a evolução da quantidade e importância dos dados armazenados nas organizações surgiu a necessidade de um sistema cuja função é manter as informações e torná-las disponíveis quando solicitadas, a esse processo denominou-se Sistema de Banco de Dados, e esses são projetados para controlar grandes volumes de informações, garantir segurança, integridade e agilidade na recuperação de dados.

De acordo com a literatura existente sobre o assunto há várias definições para banco de dados. Segundo Date (1991,p.3):

Um sistema de Banco de dados não é nada mais do que um sistema de manutenção de registros por computador. O próprio banco de dados pode ser considerado uma espécie de sala de arquivo eletrônica – ou seja, um depósito de um conjunto de arquivos de dados computadorizados que oferece diversos recursos ao usuário, possibilitando-lhe a realização de várias operações.

Devido a grande evolução dos bancos de dados surgiu a necessidade de incorporar ferramentas que monitorassem a manutenção, armazenamento, recuperação, entre outras funções, dando origem aos Sistemas Gerenciadores de Banco de Dados (SGBD).

Um Sistema Gerenciador de Banco de Dados serve para criar, gerenciar, controlar os acessos; as transações; a integridade para grandes quantidades de dados de forma eficiente e permitir que esses dados fiquem guardados e a disposição por um longo tempo com segurança. Esses sistemas são constituídos por um conjunto de dados associados a um conjunto de programas para acesso a esses dados. Um SGBD apresenta inúmeras vantagens (SILBERSCHATZ, 1999):

Consistência de dados: A consistência permite uma padronização dos dados a serem armazenados. Por exemplo, pode-se ter informações de uma pessoa como o nome, o endereço, tanto no cadastro da mesma, quanto em um cadastro qualquer que envolva essa pessoa. O SGBD permite que se possa projetar um relacionamento entre esses cadastros (tabelas) criando com isso, consistência nos dados, e evitando a redundância (duplicidade nas informações) ou manutenção de um cadastro único;

Facilidade no acesso aos dados: Um SGBD facilita o acesso aos dados, de modo que, quando solicitado para fazer uma recuperação, de acordo com o pedido do usuário, deve retornar com agilidade e exatidão a solicitação do mesmo;

Integridade: Com a integridade faz-se a manutenção da consistência dos dados, e ela assegura que os dados do banco de dados sejam corretos. Pode-se citar como exemplo o “cadastro de pessoas”, ou seja, no momento em que for cadastrado o endereço, o sistema só deixa cadastrar o bairro referente à cidade do usuário, impossibilitando com isso a inserção de um bairro (registro) de uma outra cidade;

Segurança: A segurança é uma das principais vantagens de um SGBD. Cita-se como exemplo uma “organização”, em que há dados que só um tipo de usuário pode acessar, o SGBD permite que sejam criados perfis restringindo o acesso e possibilitando que os usuários acessem somente os dados correspondentes aos seus perfis. O perfil é caracterizado pelo grupo de usuários com permissões para acessar e efetuar transações distintas nos dados;

Compartilhamento de dados: Em um SGBD as transações são executadas em isolamento, ou seja, é assegurado que o efeito de qualquer transação completa seja preservada, ainda que o sistema falhe. Para que isso aconteça, os Sistemas Gerenciadores de Banco de Dados (SGBDs) executam *backup* e *recovery* automaticamente, quando necessário, através de um gerenciador de controle de concorrência, o qual gerencia a concorrência entre as tabelas e de um gerenciador de registro de *log* e recuperação, responsável pela durabilidade das transações.

Todas essas vantagens levaram as organizações a adotarem SGBDs para gerenciar seus dados. Com o passar do tempo e a evolução, surgiram novos sistemas de SGBDs, como por exemplo, os Sistemas Gerenciadores de Bancos de Dados Distribuídos (SGBDDs), capazes de distribuir vários bancos de dados em locais diferentes, possibilitando que um lugar possa acessar dados de outro lugar, e com isso, gerenciar as transações que são atualizadas em todos os bancos, permitindo um compartilhamento dos dados entre os bancos.

A modelagem de dados tem por objetivo facilitar o entendimento das estruturas de um banco de dados. Conforme consta em Silberschatz (1999, p. 7):

na estrutura do banco de dados está o modelo de dados: um conjunto de ferramentas conceituais usadas para a descrição dos dados, relacionamentos, semântica e regras de consistência.

Usam-se com isso as propriedades do SGBD dispostas acima.

Pode-se classificar os modelos em três tipos: (SILBERSCHATZ,1999), modelos conceitual, lógico e físico. Porém, a técnica mais difundida é a de entidade-relacionamento, com esta podemos fazer o DER, que é o diagrama de entidade-relacionamento.

Modelo Conceitual: Modelo conceitual consiste na modelagem das entidades, dos relacionamentos, independente de qual SGBD irá aplicar-se à modelagem.

Modelo Lógico: O modelo lógico é a descrição do banco de dados na visão dos usuários do SGBD, tanto do ponto de vista do programador-analista, quanto do administrador do banco de dados (DBA) (HEUSER, 2000). Assim, o modelo lógico já depende do SGBD no qual se aplicará o modelo conceitual. Nesse modelo de SGBD relacional, as entidades são vistas em formas de tabelas, junto às colunas destas.

Modelo Físico: Depois de modelados os esquemas conceitual e lógico, é necessário escolher estruturas de armazenamento e caminhos de acesso específicos para os arquivos de banco de dados, de forma a alcançar um desempenho desejável para as aplicações. Essas são projetadas no modelo físico. Em Ramakrishnan(1997) consta que para criar um bom modelo físico o DBA precisa entender as requisições feitas pelos usuários para o SGBD, especialmente na criação de índices para consultas. Para que isso ocorra, o DBA deve conhecer a arquitetura do SGBD afim de projetar o modelo físico, já que cada SGBD possui uma variedade de opções para organização de arquivos e métodos de acesso (RAMAKRISHNAN, 1997). Na fase de projeto físico, o *designer* também deve escolher os tipos de dados e formatos que melhor se enquadram às especificações.

Depois de implementado, o sistema precisará de ajuste, ou seja, será monitorado o tempo de resposta, quando efetuada uma transação, a quantidade de espaço utilizado pelas estruturas e o número médio de transações que serão processadas ao mesmo tempo (MELO, 1998).

2.2 SISTEMAS DE BANCO DE DADOS DISTRIBUIDOS

Consta em Özsu(2001, p.1), que:

A tecnologia de sistemas de bancos de dados distribuídos (SBDDs) é a união daquilo que consideramos duas abordagens diametralmente opostas para processamento de dados: as tecnologias de sistemas de banco de dados e rede de computadores.

Com os bancos de dados têm-se a centralização dos dados e com a rede de computadores pode-se fazer a integração. A tecnologia dos SBDDs busca a integração sem a centralização. Özsu (2001, p.5) define bancos de dados distribuídos como uma “coleção de vários bancos de dados logicamente inter-relacionados, distribuídos por uma rede de computadores”.

Além disso, cada banco é independente, ou seja, possui sua própria função de *Database Administrator* (DBA), um servidor próprio para armazenar os dados e tornar os relacionamentos entre os bancos invisíveis para o usuário.

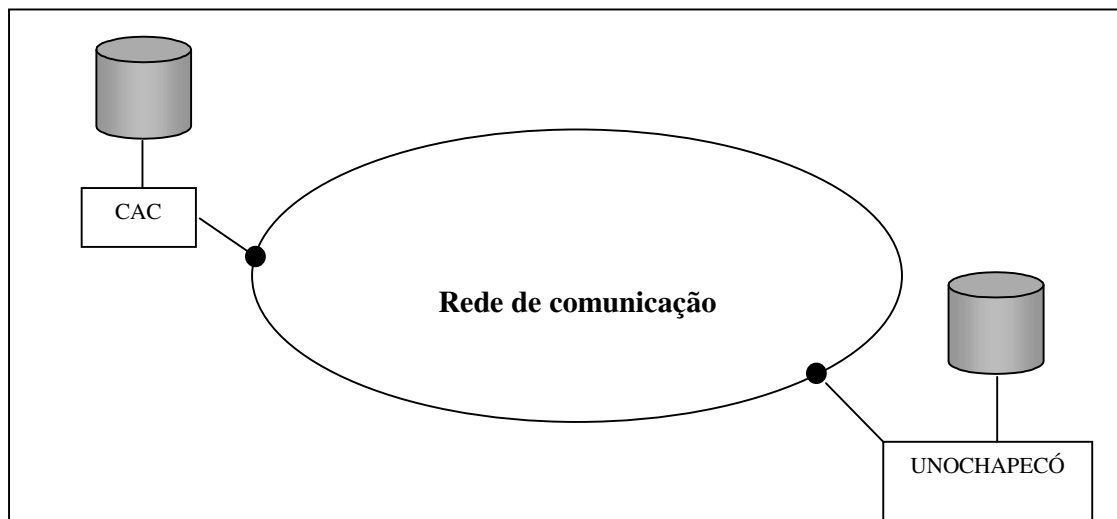


Figura 1. Exemplo de um sistema de banco de dados distribuídos

A figura 1 demonstra graficamente um sistema de banco de dados distribuídos, entre a UNOCHAPECÓ e o CAC, exemplificando o propósito deste trabalho. Percebe-se que cada localidade tem seu próprio BD, porém, um usuário do CAC pode acessar dados do CAC e também, se necessário, dados da UNOCHAPECÓ e vice-versa. Cada localidade possui seus registros, mas, as duas juntas formam um único banco de dados. O SGBD através de seus recursos mantém a integridade de cada BD local e do BD geral da UNOCHAPECÓ.

Quando mencionado que tudo isso deve ser invisível para o usuário, é para que haja uma transparência de rede, pois essa é uma das características de um sistema de banco de dados distribuídos. Com o decorrer deste capítulo será demonstrado as características, vantagens e desvantagens de um sistema de banco de dados distribuídos.

2.2.1 INDEPENDÊNCIA DE DADOS

Segundo Özsu (2001, p.10), a independência de dados é uma forma fundamental de transparência que é procurada em um SGBD. Ele afirma que “é o único tipo de transparência importante dentro do contexto de um SGBD centralizado”. A independência dos dados refere-se à imunidade de aplicativos do usuário em relação às mudanças na definição e organização de dados, e vice-versa.

2.2.2 TRANSPARÊNCIA DE REDE

Em um ambiente de gerenciamento de banco de dados distribuídos, se possível, deve-se tentar ocultar a existência da rede para o usuário. A transparência de rede caracteriza-se por proporcionar ao usuário uma visão de que os dados estejam todos em um só local, sendo que, quando usuário fizer uma solicitação ao sistema ele não precise informar o local no qual estão armazenados os dados.

2.2.3 TRANSPARÊNCIA DE REPLICAÇÃO

A replicação dos dados ocorrerá por razões de desempenho, confiabilidade e disponibilidade. A transparência de replicação refere-se ao fato de que os usuários não devam notar que o

SGBDD está sendo replicado em um outro local. A transparência de replicação deve ser uma característica padrão em SGBDs.

2.2.4 TRANSPARÊNCIA DE FRAGMENTAÇÃO

A fragmentação consiste em dividir as relações (tabelas) em fragmentos menores e tratar cada fragmento como uma relação de um banco de dados separado. As técnicas de fragmentação serão conceituadas mais adiante. Uma vantagem da fragmentação é referente a replicação dos dados, pois só é replicada parte da relação, gerando menos tráfego na rede.

2.3 PROJETO DE BANCO DE DADOS DISTRIBUÍDOS

Sobre o projeto de banco de dados distribuídos Özsü (2001,p.22) faz a seguinte abordagem: “como o banco de dados e os aplicativos que atuam sobre ele devem ser distribuídos” ele também proporciona duas alternativas básicas para posicionar os dados, particioná-los ou replicá-los. No modo de particionamento o banco de dados divide-se em vários fragmentos, os quais são colocados em um local diferente da rede. Os projetos de replicação podem ser totalmente replicados, ou seja, replicar o banco inteiro ou, parcialmente, em que cada partição do BD é armazenado em um local da rede, assim sendo, o projeto pode tornar-se particionado e replicado ao mesmo tempo. Uma questão fundamental para o projeto é a fragmentação, ou seja, a separação do BD em partições.

2.3.1 PROCESSAMENTO DISTRIBUÍDO DE CONSULTAS

Por ser uma questão importante para o desempenho, deve-se ter maior atenção no processamento de consultas em banco de dados distribuídos, portanto é importante observar dois aspectos, o envio da mensagem e a otimização da consulta, pois quando o usuário solicita uma consulta, o SGBD procurará nos locais em que estão os BDs.

Por exemplo, um usuário do CAC solicita uma consulta de dados que não estejam em seu BD local, então ele deve enviar uma mensagem solicitando ao SGBD da UNOCHAPECÓ, o que irá retornar os dados para o usuário do CAC. Esse exemplo parece simples, pois o banco está distribuído somente em dois locais, mas se o sistema estiver distribuído em três, quatro, cinco ou mais locais, perceber-se que o problema é mais complexo do que se imagina. Outro problema para os SGBDDs, é que, além de gerenciar as consultas em diferentes locais, ele precisa administrar uma possível fragmentação das tabelas.

Com essa breve explicação nota-se que a otimização em sistemas distribuídos é muito mais complexa do que em um sistema centralizado. Mas, quem gerencia o processamento de consultas é o SGBD, tornando isso um problema muito maior para os fabricantes do que para os usuários.

2.3.2 CONTROLE DISTRIBUÍDO DA CONCORRÊNCIA

O controle da concorrência deve criar uma sincronia entre os acessos aos dados, de forma que, mantenha-se a integridade e a consistência de várias cópias que estão distribuídas ao longo da rede. Uma das formas que pode ser usada para gerenciar a concorrência é o sistema de bloqueio, que bloqueia os registros enquanto é executada uma transação, e o timbre de hora, que organiza as transações em ordem para depois executá-las. Essa ordenação acontece porque são atribuídos timbres de horas às transações e aos itens de dados que estão armazenados.

O gerenciamento de bloqueios ocorre de três forma Özsü(2001):

- 1- **Bloqueio centralizado** - no qual as tabelas de bloqueio são armazenadas em apenas uma máquina da rede e esta fica responsável por gerenciar os bloqueios para as transações;
- 2- **Bloqueio de cópia primária** - se o banco está distribuído em vários locais, um dos locais fica bloqueado para acesso e depois é atualizado. (ÖZSU, 2001, p.327) cita o seguinte exemplo: “se a unidade de bloqueio x é replicada nos seguintes locais 1, 2 e 3, um desses locais (digamos, local_1) é selecionado como o site primário para x. Todas as transações que desejam acesso a x obtêm seu bloqueio no local_1 antes de poderem acessar uma cópia de x”. Se o banco não for replicado a responsabilidade de bloqueio fica distribuída entre os diversos locais;
- 3- **Bloqueio descentralizado** - neste caso a responsabilidade pelos bloqueios é compartilhada por todos os locais da rede, baseando-se no exemplo de ÖZSU (2001, p.327) citado acima, “as entidades que acessam x devem obter bloqueios em todos os três locais”.

Como o controle da concorrência é um fator de grande relevância para fatores de complicação, alguns SGBDs como o caso do Oracle®, têm um mecanismo chamado de *commit* em duas fases, o qual consiste da fase de preparação, em que todos os nós da rede são notificados de que haverá um *commit*, ou um *rollback* das transações, e a fase do *commit*. Se não ocorrer nenhuma falha, todos os sites fazem *commit* em suas transações, caso contrário, é executada a operação de *rollback* em todos os sites com suas respectivas transações. Se ocorrer falha somente em um nó da rede a operação de *rollback* será disparada automaticamente.

O controle de concorrência é um aspecto de fundamental importância, pois ele deve assegurar que no caso de falhas de hardware ou software, será mantida a consistência e a integridade dos dados.

2.3.3 FRAGMENTAÇÃO HORIZONTAL

A fragmentação horizontal consiste em particionar uma relação em suas tuplas, formando subconjunto de tuplas como mostra o exemplo abaixo:

CIDADAO 1

CODCIDADA0	NOME	CPF	TELEFONE	CODSETOR
1	Paulo da Silva	014.345.678-99	354-9427	1
2	João Medeiros	026.987.123-00	351-1406	2
3	Cristiano Santos	987.487.321-57	515-7543	1
4	Luis Fernando	354.881.019-14	182-0021	3

Figura 2 Relação que contém dados como base para exemplo de fragmentação horizontal

Partindo da relação acima, pode-se dividi-la em fragmentos a partir de suas tuplas:

CIDADA01				
CODCIDADA0	NOME	CPF	TELEFONE	CODSETOR
1	Paulo da Silva	014.345.678-99	354-9427	1
2	João Medeiros	026.987.123-00	351-1406	2

CIDADA02				
CODCIDADA0	NOME	CPF	TELEFONE	CODSETOR
3	Maria Santos	987.487.321-57	515-7543	1
4	Luis Fernando	354.881.019-14	182-0021	3

Figura 3. Representação gráfica da fragmentação horizontal

Uma fragmentação horizontal pode ser primária ou derivada. A fragmentação horizontal primária é executada com o uso de predicados definidos sobre a relação e a derivada o particionamento é obtido através de uma relação que resulta da definição de predicados sobre outra relação.

2.3.4 FRAGMENTAÇÃO VERTICAL

A fragmentação vertical é definida de acordo com as aplicações que os usuários estão usando, sendo que ela consiste em particionar uma relação em seus atributos, como o exemplo abaixo:

CIDADA0			
CODCIDADA0	NOME	CPF	TELEFONE
1	Paulo da Silva	014.345.678-99	354-9427
2	João Medeiros	026.987.123-00	351-1406
3	Maria Santos	987.487.321-57	515-7543
4	Luis Fernando	354.881.019-14	182-0021

Figura 4. Relação que contém dados como base para exemplo de fragmentação vertical

Partindo novamente da relação cidadao, pode-se obter o seguinte fragmento:

CIDADA01		
CODCIDADA0	NOME	CPF
1	Paulo da Silva	014.345.678-99
2	João Medeiros	026.987.123-00
3	Maria Santos	987.487.321-57
4	Luis Fernando	354.881.019-14

CIDADA02	
CODCIDADA0	TELEFONE
1	354-9427
2	351-1406
3	515-7543
4	182-0021

Figura 5. Representação gráfica da fragmentação horizontal

Nota-se que nesse caso foi feita uma partição criando um fragmento com os atributos CODCIDADA0, NOME, CPF, mas poderia ser feito um particionamento somente com CODCIDADA0 e CPF, por exemplo. No caso da fragmentação vertical, a relação resultante

deve ser composta por todas as chaves primárias da relação proprietária, mais os atributos que serão necessários, de acordo com a aplicação.

2.4 TÉCNICAS DE REPLICAÇÃO UTILIZADAS

Replicação de dados é a criação e a manutenção de cópias de uma base de dados ou sistema de arquivos. Estas cópias são mantidas em servidores independentes para aumentar a confiabilidade e garantir acesso local (BURETTA, 1997). A replicação de dados em sistemas distribuídos é utilizada para torná-los mais confiáveis e seguros, pois o sistema pode sobreviver a falhas de qualquer componente replicado. Além disso, todo esse processo deve ser transparente ao usuário.

Utilizando a replicação, os dados compartilhados podem ser distribuídos automaticamente para as plataformas de destino, diminuindo o tráfego na rede, gerando conseqüentemente uma disponibilidade dos dados e aumentando a performance no acesso à eles.

2.4.1 DATABASE LINK (LINK DE BANCO DE DADOS)

Com o *database link* é possível fazer a transparência local, em termos mais técnicos, um *database link* define uma conexão de um banco de dados para outro, e essa definição é armazenada no dicionário do Oracle®. Desde que exista acesso ao banco de dados remoto, pode-se ter o controle completo sobre os privilégios do banco. O *database link* faz com que os objetos remotos apareçam para os usuários conectados e com os devidos privilégios de acesso aos objetos como se fossem objetos locais (DYE, 1999, p.13).

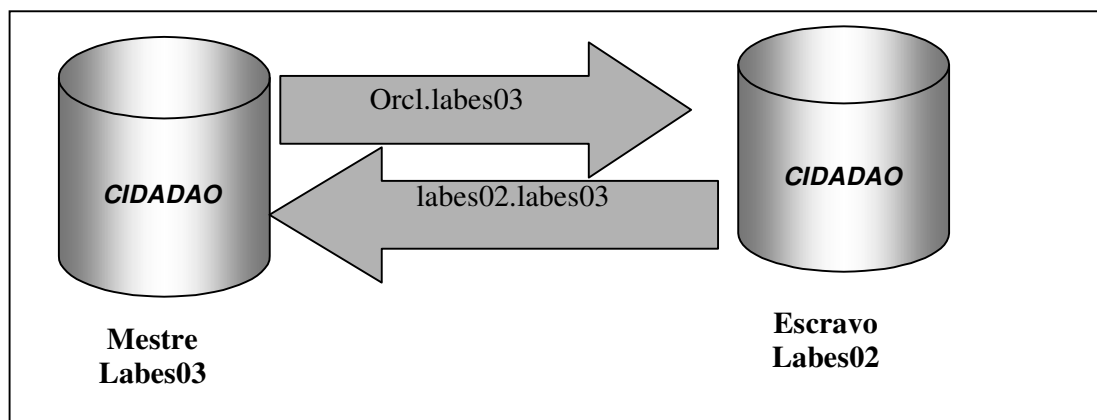


Figura 6. Representação gráfica do *database link*.

2.4.2 UPDATEABLE SNAPSHOT (VISÕES ATUALIZÁVEIS)

O *updateable snapshot* oferece um meio de disponibilizar cópias atualizáveis de dados em múltiplos lugares da rede. Assim como o *read-only snapshot*, o *updateable snapshot* deve ser atualizado periodicamente para refletirem as mudanças que ocorreram na tabela mestre, mas também pode propagar as mudanças ocorridas no *snapshot* de volta para a tabela mestre. Para isso, utiliza-se um gatilho na tabela de *snapshot* de arquivos de transição que está definido no site mestre quando o *snapshot* é reatualizado, consolidando os dados de vários BDs dentro de um único local. Essa propagação pode ocorrer ao mesmo tempo que o *snapshot* é atualizado ou em um intervalo de tempo agendado. (DYE, 1999, p.15)

O SGBD Oracle® suporta diferentes tipos de atualizações, completa e rápida, bem como atualizações manual e automática.

Atualização completa: Para executar uma atualização completa de um snapshot, o servidor que gerencia o snapshot executa a query definidora do snapshot. O resultado da query substitui o dado existente no snapshot para atualizar o snapshot. Oracle pode executar uma atualização completa para qualquer snapshot.

Atualização rápida: Para executar uma atualização rápida, o servidor que gerencia o snapshot primeiro identifica as mudanças que ocorreram no mestre desde a mais recente atualização do snapshot e então as aplica para o snapshot. Atualizações rápidas são mais eficientes do que completas quando existem poucas alterações para o mestre porque servidores participantes e rede replicam menos dados.

Atualização automática: Quando se cria um grupo de snapshot atualizável, administradores usualmente configuram o grupo então o Oracle® automaticamente atualiza seus snapshots. De outro modo, administradores teriam condições de realizar atualização manualmente quando necessário.

Atualização manual: Programadas, atualizações automáticas de snapshot podem não ser adequadas. Por exemplo, imediatamente acompanhando um volume de dado carregado em uma tabela mestre, snapshots dependentes não mais representará o dado da tabela mestre. Ao invés de esperar pela próxima programação automática de atualizações de grupo, você deve querer manualmente atualizar grupos de snapshot dependentes para imediatamente propagar as novas linhas de uma tabela mestre para snapshots associados.

Duas importantes características do *updateable snapshot* em qual ele se distingue das tabelas replicadas, é que eles atualizam somente o site mestre e podem ser desconectados do site mestre por um longo período. Além disso, ela suporta a técnica de fragmentação horizontal. Por outro lado, os *updateable snapshot* também possuem algumas restrições. Entre as quais pode-se citar: (DYE, 1999, p.306)

- Devem ser *snapshots* simples, ou seja, devem ser feitos em uma única tabela sem *DISTINCT*, *GROUP BY*, ou subconsultas;
- Não suportam colunas do tipo *LONG* ou *LONG RAW*;
- Devem conter todos os atributos da tabela mestre, deste modo a fragmentação vertical não é possível.

Pode-se obter uma *updateable snapshot* com os seguintes comandos:

```
CREATE MATERIALIZED VIEW cac.cidadao BUILD IMMEDIATE REFRESH
FORCE START WITH to_date('29/09/2003 12:21:53','dd/mm/yyyy HH24:MI:SS')
NEXT sysdate + 1/1440 FOR UPDATE AS
SELECT COD_CIDADA0,NOM_CIDADA0,DATA_CADASTRO,ESTADO_CIVIL,
SEXO,CPF,RG, ORGAO_EXP_RG,DATA_EXP_RG, CGC,
NATURALIDADE,NOM_PAI, NOM_MAE,GRAU_ESCOLARIDADE,
COD_PROFISSAO,OBSERVACAO
FROM CIDADA0@orcl.labes03
WHERE CIDADA0.COD_CIDADA0 BETWEEN 70000000 AND 80000000 OR
CIDADA0.COD_CIDADA0 BETWEEN 1 AND 20000 OR
CIDADA0.CODCURSO IN (1010,1023);
```

Nota-se que neste exemplo, cria-se uma visão atualizável, com fragmentação de tabela, sendo que a fragmentação foi obtida através da clausula where imposta. A seguir um exemplo de visão atualizável, onde não foi usada a técnica de fragmentação:

```
CREATE MATERIALIZED VIEW cac.ARQUIVO BUILD IMMEDIATE REFRESH
FORCE START WITH to_date(' 29/09/2003 12:21:53' , ' dd/mm/yyyy HH24:MI:SS' )
NEXT sysdate + 1/1440 FOR UPDATE AS
SELECT ARQUIVO.* FROM ARQUIVO@orcl.labes03;
```

Neste caso foi criada a visão, sem fragmentação, ou seja, ela foi replicada inteira. Após a criação da visão faz-se necessário coloca-las em um grupo de atualização, onde ele é obtido através do seguinte comando:

```
BEGIN
  DBMS_REPCAT.CREATE_SNAPSHOT_REPOBJECT(
    gname => ' "GRP_CAC_MESTRE"' ,
    sname => ' "CAC"' ,
    oname => ' "CIDADA0"' ,
    type => ' SNAPSHOT' ,
    min_communication => TRUE);
END;/
```

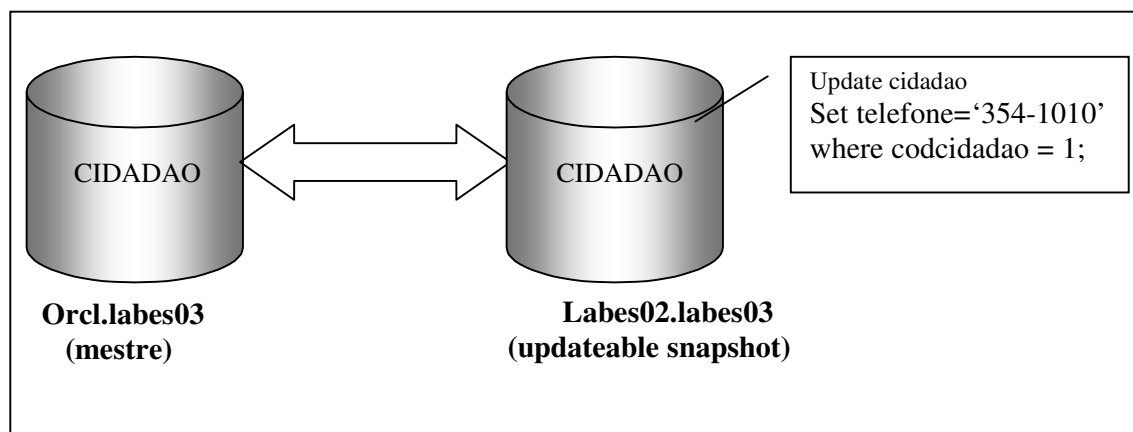


Figura 7. Representação gráfica do visão atualizável.

2.6 PROTÓTIPO

2.6.1 AMBIENTE CAC

No momento atual, nota-se uma grande deficiência de hardware existe no CAC. Porque além dos equipamentos clientes, que não vem ao caso para a utilização do sistema, percebe-se uma deficiência de servidor, sendo que hoje o CAC comunica-se com a UNOCHAPECÓ através um link de ADSL empresarial de 128KB, em conjunto com uma VPN de 128 bits de criptografia, tudo isso em um servidor constituído de um micro computador AMD K6 – 500 Mhrz, com 128 MB de memória RAM e com 20 GB de espaço para ser utilizado no disco rígido. O sistema operacional usado até então é Linux da espécie Red Hat 7.3.

O único problema com a ADSL é que nem sempre tem toda a banda disponível e em horários de pico perde muita performance, mas o maior problema está no servidor, que como descrito acima, é uma máquina de pequeno porte, não sendo ideal para ser um servidor de banco de dados. Como vai ser utilizado o SGBD Oracle®, sugiro que seja adquirida uma máquina

mais robusta, para não perder performance dentro da rede do CAC, bem como garantir a segurança dos dados.

Nesse primeiro momento, percebe-se que pode ser utilizada uma técnica de replicação assíncrona, sendo que pode haver um grande tempo de latência para que os dados sejam atualizados tanto no banco de dados existentes no CAC, como o existente na UNOCHAPECÓ.

2.6.2 POR QUE DISTRIBUIR OS DADOS?

Apesar de todas as desvantagens de se ter um sistema de banco de dados distribuídos, como foram descritas em capítulos anteriores deste trabalho, faz-se necessário a distribuição dos dados para além de obter um ganho desejável de performance dentro da rede interna do CAC, o sistema vai ficar livre de qualquer problema que venha à ocorrer, provocando a desconexão de seu link com a UNOCHAPECÓ, visto que a replicação dos dados vai ter um grande tempo de latência. Essa autonomia provê total independência para os dados nas operações do dia a dia, isso acontece em virtude do servidor local possuir réplicas atualizadas dos dados, mesmo que ocorra uma interrupção na comunicação entre o site mestre e o servidor de réplicas.

A replicação dos dados vai ocorrer três vezes por dia, ou seja, se acontecer qualquer problema na empresa que fornece o link de ADSL, o CAC vai funcionar normalmente, sendo que os dados que seu sistema necessita estão todos disponíveis e as devidas atualizações são utilizadas somente em dias posteriores, visto que todos os atendimentos efetuados pelo CAC em benefício da comunidade são previamente agendados.

Outro aspecto de relevância para distribuir o banco de dados é em relação ao custo. Dependendo do cliente, por exemplo, é mais barato comprar uma licença do SGBD Oracle® Standard do que manter o link da ADSL por um ano, a aquisição de um cliente Oracle®, pode se tornar 30% mais barata no primeiro ano, e como adquire-se a licença para sempre, esse custo vai baixando ao decorrer de um período.

Aplicando todas as regras necessárias para a distribuição de dados, o sistema vai ser totalmente independente, podendo operar com suas transações normalmente com base nas tabelas replicadas, assim garantindo a confiabilidade do sistema, de tal forma que deve haver uma atualização completa dos dados (replicação), mantendo a integridade e consistência de todos os bancos distribuídos ao longo da rede.

Ao final da aplicação de todo esse conjunto de recursos, o desempenho deverá ser visivelmente otimizado em comparação com a atual configuração do sistema descrito anteriormente. Esta otimização do desempenho dar-se-á em consideração de que é possível executar mais de uma consulta ao mesmo tempo, ou desmembrar esta consulta em várias subconsultas. Como cada transação age sobre parte do banco de dados com isso a concorrência pela CPU e as operações de entrada e saída (E/S) são menos executadas. Outro aspecto que garante uma melhor performance para o sistema acontece em virtude da localização dos dados reduzindo o tempo para recuperação desses dados.

Por fim todas essas vantagens estão aliadas com a técnica de fragmentação horizontal, que como explida em capítulos anteriores, consiste em particionar as relações a partir de suas tuplas, replicando somente o fragmento que contém os dados necessários para o andamento das operações impostas ao local. Essa fragmentação é possível em algumas relações porque os dados necessários estão separados por faixa de códigos diferentes, gerando uma distinção da origem dos mesmos. Lembrando que toda a fragmentação deve ser transparente ao usuário.

2.6.3 FRAGMENTAÇÃO UTILIZADA NO PROTÓTIPO

Nesse estudo de caso, foi utilizada a técnica de fragmentação horizontal, devido ao fato de que as tabelas dispostas na modelagem em anexo (anexo II), todas elas pertencem somente ao sistema em funcionamento no CAC, com exceção das tabelas CIDADÃO, ALUNO E CURSO. Estas três relações pertencem aos dois esquemas descritos até o momento. No restante das tabelas são criadas visões atualizáveis completas (réplicas idênticas), sendo que nestas três citadas acima foi elaborada uma regra para aplicar-se a fragmentação horizontal derivada, visto que o banco deve permanecer com a integridade relacional imposta quando modelado.

A regra utilizada na relação cidadão foi através das faixas de códigos, replicando somente as pessoas cadastradas pelo CAC, que possuem um intervalo específicos para sua identificação, seus códigos estão entre 70.000.000 e 80.000.000. Outra regra é no intervalo de 1 até 20.000, este é para criar a visão de funcionários e professores. Para o caso dos alunos, a regra aplicada foi considerando o curso, como hoje existe somente atendimentos na área de Direito e Serviço Social, criou-se a visão alunos, com alunos do curso de Direito e Serviço Social, automaticamente fragmentando a tabela CURSO usando os códigos referentes aos cursos citados a cima, assim sendo criando também uma terceira regra para a relação CIDADÃO, que é replicar as pessoas referentes a estes cursos. Para este caso, tivemos que criar uma pequena redundância referente à modelagem do esquema, visto que as visões atualizáveis não podem ser criadas usando como base mais de uma tabela mestre. Para este problema ser resolvido foi acrescentado na tabela CIDADÃO um atributo para indicar o curso da pessoa, onde foi preenchido somente as pessoas que fazem os cursos que tem atendimento ao CAC (Direito e Serviço Social).

As características da completeza, disjunção e reconstrução são garantidas conforme segue :

Completeza: No caso da fragmentação horizontal derivada, é a completeza que garante a integridade referencial dos dados, como por exemplo um cidadão só vai poder ser estagiário, se ele estiver no fragmento de aluno, após a aplicação da regra de fragmentação para a tabela ALUNO, descrita acima.

Reconstrução: A reconstrução da relação original através de seus fragmentos, é obtida através da aplicação da operação de união. Fazendo a união dos fragmentos, o resultado é uma relação idêntica a original.

Disjunção: A disjunção é aplicada através das regras estabelecidas para a fragmentação. Tomando como base a relação CIDADÃO, a qual tem três critérios para a fragmentação, imaginamos que ela seria fragmentada em três partes, para garantir que a relação global está disjunta, os dados do primeiro critério que são os códigos que começam a partir 70.00.000, não podem estar no fragmento que contém os dados do segundo critério, que são os códigos entre 1 e 20.000. E esses não podem estar no terceiro fragmento que corresponde ao terceiro critério. (ÖZSU,2001)

2.7 CONCLUSÃO

Ao final deste trabalho é possível observar que o problema da distribuição dos dados em sistemas de bancos de dados não é trivial e que existem muitas variáveis e métodos que se aplicam a distribuição de dados, alocação e replicação. Alguns dos problemas e métodos relacionados na bibliografia mais difundida foram estudados, porém somente alguns foram implementados, em função da amplidão do espaço de soluções, especialmente com relação a fragmentação dos dados e também em função dos objetivos deste trabalho. A fragmentação horizontal foi realizada e as regras de correção nortearam este trabalho, especialmente com

relação a fragmentação horizontal, o único método que se fez necessário para a implementação do protótipo, também sendo consideradas as aplicações, fundamentais para o desenvolvimento do sistema. Desta forma se oferece uma solução possível para a descentralização dos dados, proporcionando a agilização dos processos burocráticos que envolvendo os setores da organização.

É possível notar, um alto nível de dificuldade quando envolvemos questões de banco de dados distribuídos, causando grandes problemas não esperados quando se está apenas projetando a distribuição e a replicação dos dados, como por exemplo as triggers de base, funções e procedimentos existentes no banco de dados mestre, devem ser todas elas recriadas no escravo. Outro aspecto de alta relevância é em relação a seqüências, como visões atualizáveis não implementam seqüências, se faz necessário um controle de integridade de chave-primária sempre que se gera códigos para chaves-primárias a partir de seqüências.

No que diz respeito ao estudo de caso realizado, pode-se concluir que o sistema que está em funcionamento, vai obter primeiramente um ganho notável de performance, mas também vai ter que lidar com problemas relacionados com a concorrência dos dados e a fragmentação das tabelas. Todos esses impasses podem ser contornados através das próprias aplicações, gerando assim uma pequena mudança nos códigos fontes dos programas. Outro aspecto a ser considerado é a replicação de visões e funções existentes no site mestre que devem duplicadas para o site escravo, em função das aplicações que rodam sobre elas. Não sendo possível replicar visões e funções, elas devem ser criadas novamente no site escravo. Quanto a função de renovação das visões atualizáveis, deve se tomar um cuidado especial, visto que qualquer problema que ocorra, as mesmas não se atualizam automaticamente, sendo assim, deve-se construir um mecanismo que gerencie os erros, ignorando-os e atualizando as visões.

Toda a técnica aplicada, foi baseada em um problema encontrado na UNOCHAPECÓ, o que faz concluir que a aplicação foi desenvolvida para atender um problema específico, provando assim que se torna possível implementar a sugestão disponibilizada na pesquisa. Esperando com isso uma diminuição no custo monetário que está sendo aplicado mensalmente para que o sistema computacional tenha pleno funcionamento, agilizando os processos de atendimento a comunidade.

Com tudo isso, este estudo ajudou a entender com maior profuncidade como funciona um SGBD relacional distribuído, algumas das dificuldades e possíveis soluções ao se projetar um sistema onde os bancos de dados estão alocados em lugares fisicamente diferentes, bem como, entender a proposta do conhecimento científica para a distribuição e replicação de dados e a tecnologia Oracle® para a implementação de SGBDDs.

REFERÊNCIAS BIBLIOGRÁFICAS

BURETTA, Marie. **Data Replication: Tools and Techniques for Managing Distributed Information**. New York: Wiley, 1997.

DATE, C. J.. **Introdução a sistemas de Banco de Dados**. Tradução da 4ª ed, Rio de Janeiro: Campus, 1991.

DYE, Charles. **Oracle distribuied systems**. Sebastopol: O' Reilly, 1999. Janeiro: Campus, 2000.

ELMASRI, Ramez, NAVATHE, Shamkant B. **Fundamentals of Database Systems**. 2.ed,USA, California, Menlo Park: Addison-Wesley Publishing Company, 1994.

HESSEL, Claudio. Universidade Federal do Rio Grande do Sul. Instituto de Informática. **ODL, OQL e Linguagem Visual para Consulta**. Disponível em :

<http://shark.inf.ufrgs.br/cmp151_ModelosBD/cmp15120001/SeminarioClaudioHessel/Artigo.doc>. Acesso em: 15 abr. 2003.

HEUSER, Carlos Alberto. **Projeto de banco de dados**. 3ª ed., Porto Alegre: Sagra Luzzato, 1999.

LONEY, Kevin, THERIAULT, Marlene. **Oracle8i O manual do DBA**. Rio de Janeiro: Campus, 2000.

MELO, Rubens N., SILVA, Sidney Dias da, TANAKA, Asterio K. **Banco de Dados em Aplicações Cliente-Servidor**. Rio de Janeiro: Livraria e Editora Infobook S.A., 1998.

ÖZSU, M. Tamer, VALDURIEZ, Patrick. **Princípios de Sistemas de Banco de Dados Distribuídos**. Tradução da 2ª ed., Rio de Janeiro: Campus, 2001.

RAMAKRISHNAN, Raghu. **Database Management Systems**. USA, Boston: McGRAW-HILL INTERNATIONAL EDITIONS, 1997.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN S.. **Sistema de Banco de Dados**. Tradução da 3ª ed., São Paulo: Makron Books, 1999.